Telecommunication Systems Journal manuscript No.
(will be inserted by the editor)

Packet loss reduction during rerouting using network traffic analysis

W. Tavernier · D. Papadimitriou · D. Colle · M. Pickavet · P. Demeester

Received: date / Accepted: date

Abstract Upon certain network events, such as node or link failures, IP routers need to update their affected routing table entries. During the period between the failure occurrence and the installation of the updated entries (on the line cards), the network traffic is lost when forwarded by routers that are still using old entries. Indeed, current IP routers do not involve network traffic information during this unordered update process. The consequence is more packet losses compared to a process that would order these entries based on local traffic information. In this paper, we model and predict network traffic passing through an IP router and define two dynamic heuristics in order to reduce the packet loss resulting from routing table updates. AutoRegressive Integrated Moving Average (ARIMA) - Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) traffic models are used in combination with heuristics that dynamically sort the routing entries and improve the low-level routing table update process. In a realistic simulation environment, we show that this setup can result into a clear decrease of packet loss, depending on i) the network traffic model, ii) the applied heuristic, and iii) the network traffic aggregation level.

Keywords routing · recovery · routing table update · traffic analysis · arima · garch · packet loss reduction

W. Tavernier · D. Colle · M. Pickavet · P. Demeester
Ghent University, IBCN-IBBT, INTEC,
Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium
E-mail: {wouter.tavernier, didier.colle, mario.pickavet, piet.demeester}@intec.ugent.be

D. Papadimitriou
Alcatel-Lucent Bell
Copernicuslaan 50, B-2018 Antwerpen, Belgium
E-mail: dimitri.papadimitriou@alcatel-lucent.be

1 Introduction

Current IP networks run dynamic routing protocols to automatically compute their routing tables. For this purpose, link-state routing protocols are commonly used in today's IP networks, and referred to as Interior Gateway Protocols (IGP). The (re-)convergence time of routing protocols can take (tens of) seconds upon topological change(s) resulting from, e.g., link or node failures. Henceforth, many techniques have been developed to help IP networks reducing the convergence time of their routing tables upon occurrence of these events. Their common target is to make sub-second (re-)convergence possible [13] in order to prevent the disruption of the traffic flows affected by the routing table updates resulting from these events.

Fast failure detection techniques ensure detection times in the order of milliseconds, for example, using hardware-based loss-of-signal detection or software-based Bidirectional Failure Detection (BFD, [15]). Recovery techniques such as plain IP Fast ReRoute (FRR, [27]) aim to minimize the duration of traffic delivery disruption caused by link or node failure(s) until the network re-converges on the new topology. Common to all these techniques is their focus on recovery time and availability of a (loop free) backup path.

While the backup path can be known (i.e. pre-calculated) at the moment of failure detection, the low-level update of routing entries at the IP level can still take more than 1 second in IP backbone networks [13]. Therefore, updating one forwarding entry before another can be of high value in order to reduce the transient losses of packets. This can be illustrated by the scenario in Figure 1. The figure presents the simplified situation of an IP backbone router B needing to forward 3 traffic flows from router A towards router D. Whereas the shortest path for these traffic flows is via router C, a failure between router B and C forces router B to update its routing table. As will be discussed in more detail

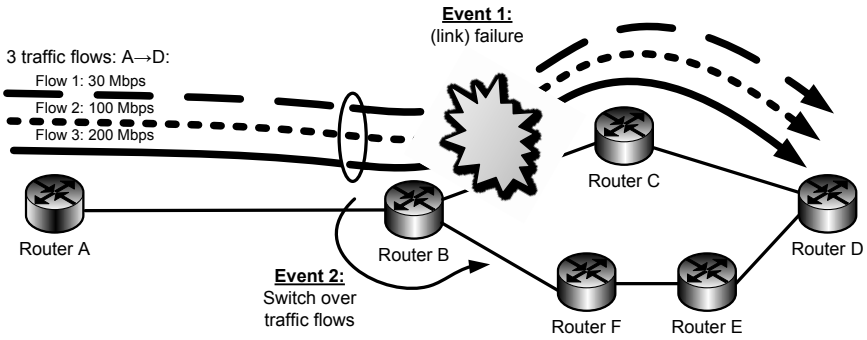


Fig. 1 IP backbone router about to update routing table entries corresponding to three traffic flows

in the next section, the default IP rerouting process foresees no specific order for the update of the routing entries corresponding to the three traffic flows. For example, updating the flows in order (1,2,3) would be perfectly viable. Because packet loss occurs as long as the entries are not updated (traffic is sent towards a black hole), 200 Mb could be lost if the routing entry corresponding to the third flow would only be updated as last in the batch after 1 second. Clearly it would be beneficial if the update order would be (3,2,1). While in this example only 3 routing table entries need to be updated, in a realistic scenario of an IP backbone network with 10 Gbps or 40 Gbps links where routers need to update thousands of routing entries, packet loss reduction techniques during rerouting events can make a big difference.

The study conducted in [29] was the first to show that, under the simplified assumption that network traffic remains stable during the routing table update, a decrease of packet loss could be obtained when changing the update order of routing table entries (see Section 3). In this paper, we show that realistic network traffic in a IP backbone network does fluctuate during sub-second periods. Therefore, more accurate network traffic pattern knowledge is needed to model and predict network traffic during the process of updating routing table entries. Given more advanced traffic models, we show that the routing table update process can be further improved on a lower process level using heuristics that dynamically calculate: i) the update order of the routing entries, and ii) the quantum time of low-level router process in order to decrease the resulting packet loss.

The paper is structured as follows. In Section 2 we describe how routed IP networks deal with network failures and investigate in more detail the default local rerouting process. The concept of traffic-informed rerouting is introduced in Section 3, containing a state-of-the-art overview and outlining the approach followed in this paper. Traffic monitoring as explained in Section 4 is the first step of the outlined process in the previous section. In Section 5 we analyze realistic network traffic traces of IP backbone routers and investigate their sub-second dynamicity. The same section then gives an overview of a set of simple and more advanced state-of-the-art models for network traffic characterization and prediction. Given the input of network traffic models, Section 6 formalizes the goal of packet loss reduction and describes two packet loss minimization heuristics usable during the traffic-informed routing table update. The resulting set of techniques is benchmarked in Section 7, quantifying the accurateness of traffic models, and measuring the resulting packet loss and recovery time consequences with respect to the used traffic aggregation levels. Section 8 recapitulates the resulting computational complexity, and a conclusion is formulated in Section 9.

2 The traffic-agnostic IP routing table update

In order to determine which processes can be improved to reduce the packet loss resulting from rerouting events, this section will describe the IP routing table mechanics in more detail. The usual IP router consists of two components: a forwarding engine and a routing engine. When a data packet arrives at an incoming interface of an IP router, the forwarding engine looks for an entry in its Forwarding Information Base (FIB) and performs a longest prefix match on the destination IP address of the incoming packet. It then forwards the traffic towards its next hop according to the best matching entry. The FIB can either be manually configured or can be populated by routing table entries computed by a routing protocol.

Routing protocols ensure that topology information is distributed over the network, or the Autonomous System (AS), such that individual routers can maintain a consistent and up-to-date full view of network topology. We focus on IP backbone routers operating Link-State (LS) Interior Gateway routing Protocols (IGP) such as the Open Shortest Path First protocol (OSPF, [19]). These flood LS protocol data units (PDUs) over the network, containing information about the local links and MA (multi-access) networks a router is connected to. All received LS PDUs are

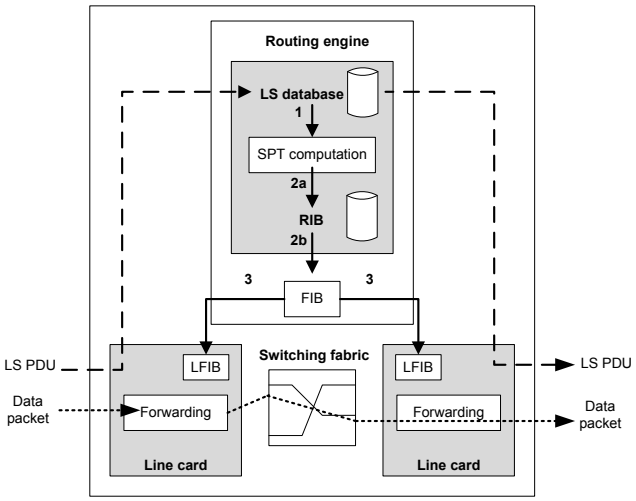


Fig. 2 The router update process

collected into a database (*the LS database*) which allows a router to have a complete view on the network link topology and to calculate shortest paths towards different destinations (IP addresses) or network parts (IP network prefixes). The LS database is updated by either detecting a local connectivity change (e.g. failing link or interface), or by receiving an LS PDU from a peering router.

A router detecting a link failure originates a LS Update (LSU) message. This message, containing LS Advertisement(s) (LSA) which describe the topological link state change(s), is reliably disseminated in the network (flooding). At every router receiving the LSU, the following three-step process executes (see Figure 2):

1. Re-computation of the *shortest path tree* (1) using the topological information stored in the updated LS DataBase (LSDB);
2. Update of the central *Routing Information Base (RIB)* and the central *Forwarding Information Base (FIB)* based on the Shortest Path Tree (SPT) computation (2a and 2b).
3. *Distribution of central FIB* towards the local FIB (LFIB) on line cards (3).

The SPT is usually re-computed in its entirety and takes about 30 to 50 μs per IGP destination prefix. Time optimizations can be obtained using incremental SPF (iSPF) algorithms [18,20]. The second step consists of updating the central RIB and FIB, using the calculated shortest paths. This uses about 50 to 100 μs per destination prefix [13]. Typically, this step happens in (pseudo-)parallel with step 3, which is about distributing the central FIB entries towards the line cards' LFIB. Running step 2 and 3 in (pseudo-)parallel, means that they both use the central CPU in interleaved time slots, swapping between both processes for updating and distribution. This process can be compared to the usual process scheduling in time-sharing Operating Systems (OS) such as Linux, whereas commercial routers make use of a hard real time OS. The consecutive time the central CPU is spending to perform central RIB/FIB entries update(s) or distribution of FIB entries towards the line card(s) is determined by the quantum of the swapping process. The quantum time can typically be configured between 10 and 800 ms. Similar quantum time values were found in [13]. In practice, the local re-convergence results into a series of update-distribution batches, where during a first time quantum a fixed set of prefixes are updated towards the central RIB/FIB, followed by a time quantum during which the same set of prefixes is distributed towards the LFIBs.

By default, the update of RIB/FIB entries (i.e., IGP prefixes) and their distribution are not ordered. Moreover, the size of the batches is determined by the quantum time of the router process, usually a preconfigured constant number over the entire update process. In this context, the proposed technique aims to sort the routing entries in decreasing bit rate order at update time while taking into account the size of the update-distribution batches.

3 Traffic-informed rerouting

3.1 State-of-the-art

The previous section illustrated that traditional IP routers are currently not designed to efficiently handle the scenario depicted in Figure 1. However, at first sight, several relatively simple approaches may be followed. For example, one might assign priorities to routing table entries corresponding to IP destination prefixes, as implemented by some vendors [3]. The resulting priorities then lead to a predefined update order of the routing table entries. While this may be feasible for small networks, given that the network operators have sufficient knowledge about the spatio-temporal properties of the traffic transported by their networks, it is definitely not practical or scalable for larger IP backbone networks, as it requires manual intervention.

An alternative approach to the problem was formulated in [29]. The main idea of this work relies on ensuring that routing entries corresponding to higher bit rate traffic should be updated and distributed (from the RIB towards the local FIB) earlier than those corresponding to lower bit rate traffic. However, the study was performed using generated network traffic from Pareto distributions with the assumption that network traffic behaves persistently (stable or constant traffic) during the routing table update process (which can take more than 1 second). As we will show in the next section, this assumption does not hold for real network traffic, leading to the need for more sophisticated traffic models and packet loss reduction heuristics.

3.2 Suggested approach

In order to thoroughly tackle the problem of traffic-informed routing table updates, we suggest the inclusion of the following router components (Figure 3):

- a *monitoring component* to maintain traffic volume and rate statistics associated to different IP prefixes (routing entries)
- a *traffic modeling component* to characterize and estimate the traffic volume and rate associated to IP prefixes (routing entries)
- a *packet loss reduction component* to configure the resulting order and size of update batches upon recalculation (see Section 2).

Each of components will be discussed in larger detail in the following sections.

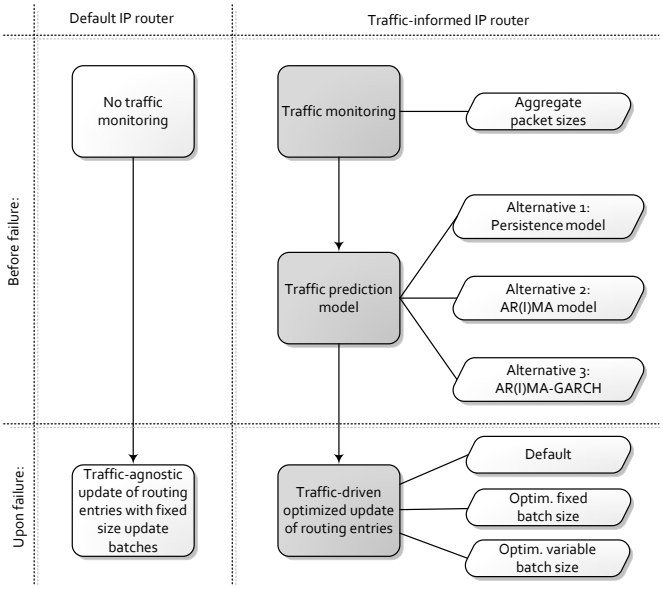


Fig. 3 Default IP router vs. a traffic-informed IP router

4 Traffic monitoring

Given the evolution of routers' line cards in terms of high bit rate interfaces (including transmitters/receivers, framers, etc.) and fast packet processors (on-board high frequency CPUs), real-time network traffic monitoring per IP prefix (corresponding to a routing entry) has become feasible [1,5, 2]. As previously motivated, obtaining statistics of the traffic properties can be of high value during the event of router updates. The following methods exist to gather data such as to estimate the traffic volume associated to an IP destination prefix at failure detection time:

- *Online statistical counters* measure aspects of transiting traffic in a router using counters, for example the number of packets per destination prefix or used packet size distribution curves (for example [4]).
- *Traffic sampling* by means of samplers. Instead of counting certain traffic characteristics, unmodified traffic is captured for some time interval. This sample is then used to derive certain characteristics, for example done in Netflow-alike settings (e.g. [12]).

As we will not focus on the metrology problem, we redirect the reader to the above-mentioned references for more detailed information. However we assume the following running conditions. We assume the monitoring system to be *passive* in the sense that no additional traffic is inserted into the network for the sake of monitoring[1]. In fact, active monitoring wouldn't be of any specific help, because we don't measure the liveness of the forwarding path, or the available bandwidth on the forwarding path.

We expect the monitoring system to be *independent* and *distributed*, meaning that we do not assume any form of centralized monitoring entity. Next, it is our assumption that the monitoring system is based on an *open loop*, meaning that the counts or estimations do not lead to actions modifying the corresponding values.

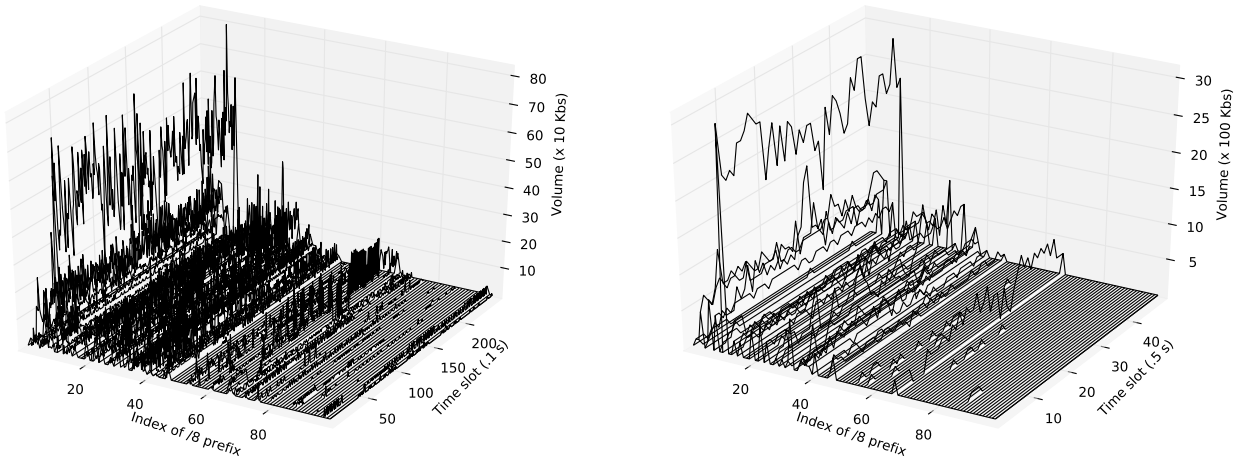
5 Network traffic analysis

Using network monitoring techniques, the volume of traffic per IP destination prefix (corresponding to the routing table entry) can be measured for a given time interval. Aggregating traffic arriving within a given time interval or time bin is referred to as *temporal traffic aggregation*. Larger bin sizes lead to more aggregation and typically smoother traffic curves, as may be observed in Figure 4a. Another level of traffic aggregation can be applied at the subnet level, by taking packets whose destination IP addresses belong to the same subnet together (smaller subnets result into more aggregation). This is referred as *spatial aggregation*.

5.1 Network traffic persistence

One might assume, as in previous work (see Section 5.2.1), that network traffic in an IP backbone network remains stable (persistent) during short (sub-second) time periods. To investigate the validity of this persistence assumption, we

[1] *Active monitoring* techniques wouldn't be of any help with respect to the characterization of the consumed bandwidth per destination prefix, as they typically check the available bandwidth or the liveness of an individual forwarding path. However, for the specific goal of detecting link failures (for triggering the rerouting process), detection techniques such as BFD can be considered as active measurements, as they introduce *Hello messages* to check the liveness of a link.



(a) Aggregating traffic using 100 ms time bins

(b) Aggregating traffic using 500 ms time bins

Fig. 4 Aggregation

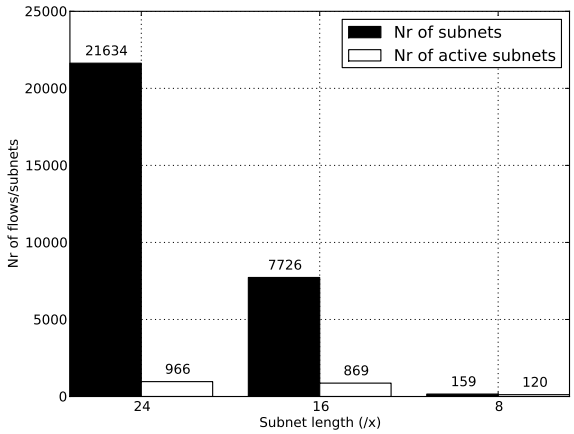


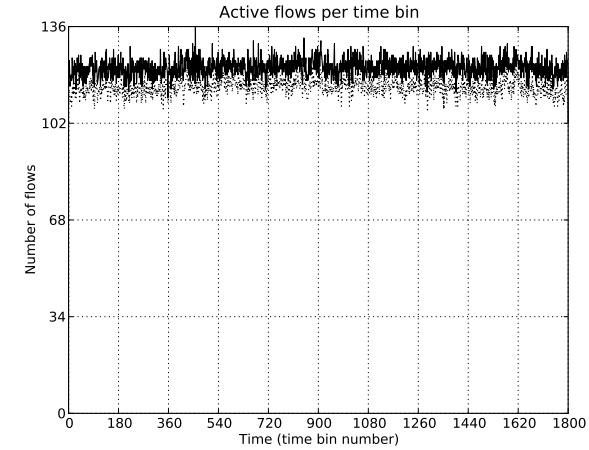
Fig. 5 Number of flows at different spatial aggregation levels

analyzed a set of trace files obtained from the Samplepoint F monitoring point deployed in WIDE backbone network (related to the MAWI project [9]). Samplepoint F monitors traffic on a trans-Pacific line (150 Mbps link) that is in operation since July 1st, 2006. We analyzed trace files at several aggregation levels. We inspected the files at the level of /8, /16 or /24 IPv4 subnet prefixes (spatial aggregation), using time intervals of either 100, 500 and 1000 ms time bins (temporal aggregation). This results into 9 aggregation level combinations.

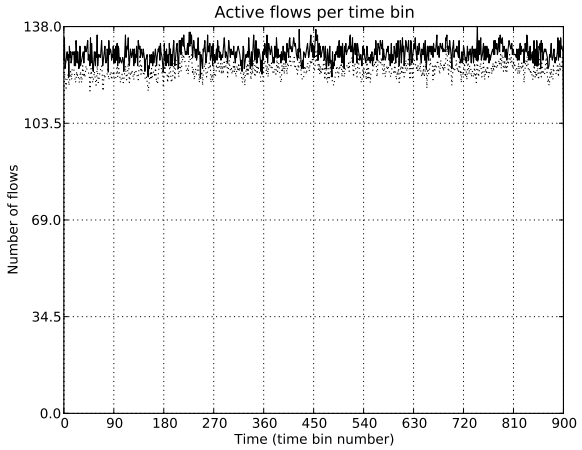
The resulting analysis sets were generated by a preprocessing module generating a two-dimensional table which groups all accumulated packet sizes per subnet for each of the above time intervals. One cell in the table corresponds to a group of packets for the given destination prefix within a given time bin. This resulted into: on average 21K /24 subnets, 8K /16 subnets, and 0.16K /8 subnets for a 15 minute trace containing on average 30 million packets (also see Figure 5).

The persistence analysis using time bins of 0.5 s and 1 s, as shown in Figure 6, shows two important aspects: i) the of number of active flows, and ii) the number of persistent flows. A flow is considered as active, if a non-zero volume is associated to it in the respective time bin. The number of persistent flows refers to the number of active flows that remain active in the next time bin. These results illustrate that the global activity and persistence level, i.e., the number of flows, is more or less stable over the duration of the traces. However, depending on the aggregation granularity, the persistence level is lower than the activity level. At coarser aggregation levels (using for example /8 subnet prefixes), the persistence-level is close to the activity-level. However, the persistence level for /16 or /24 subnet prefixes is significantly lower (only about 50 to 60 percent of the active flows stays active the next time interval). This high churn rate gives an indication on the high variability of network traffic on a sub-second timescale.

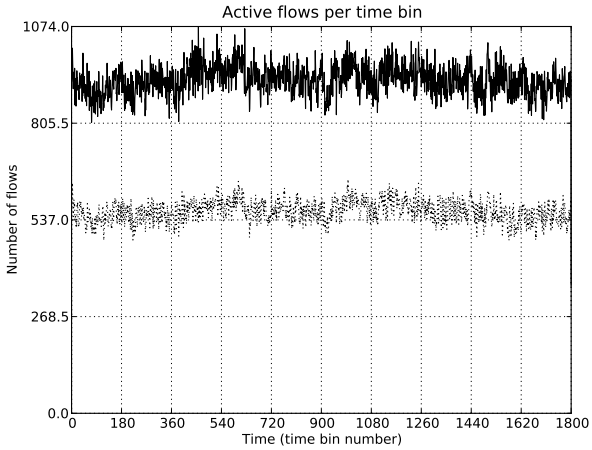
The above observations show that persistence at small timescale may not be assumed, implying that simply using a *persistence traffic model* (as done in [29]) is probably too optimistic for the goal we want to achieve. This is the direct trigger to evaluate more advanced traffic models in the coming sections. This observation does not seem to hold at larger time scales. Several studies such as [31] have shown that large flows (elephants) and small flows (mice) seem to behave more predictable and persistent over larger time scales (larger than one second).



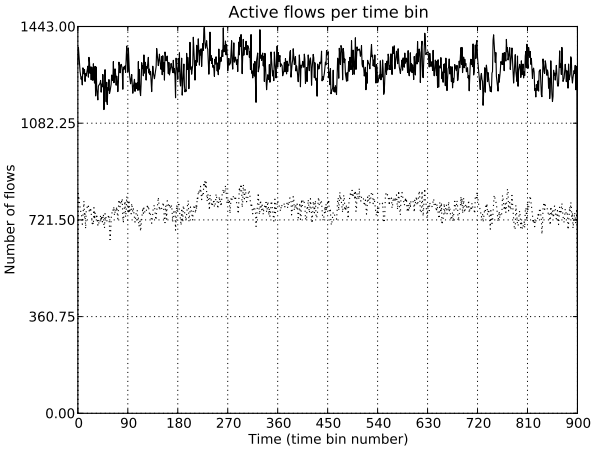
(a) /8 subnet-500 ms



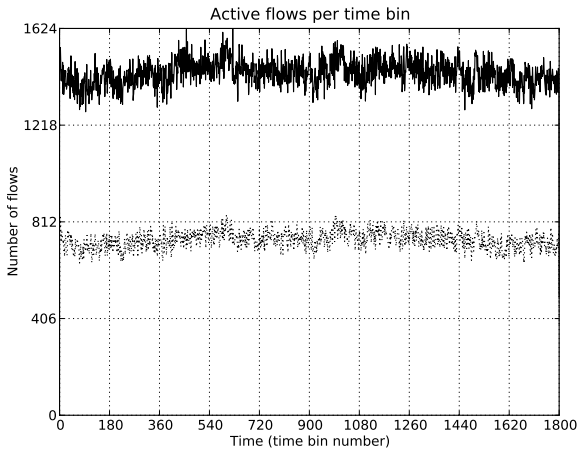
(b) /8 subnet-1000 ms



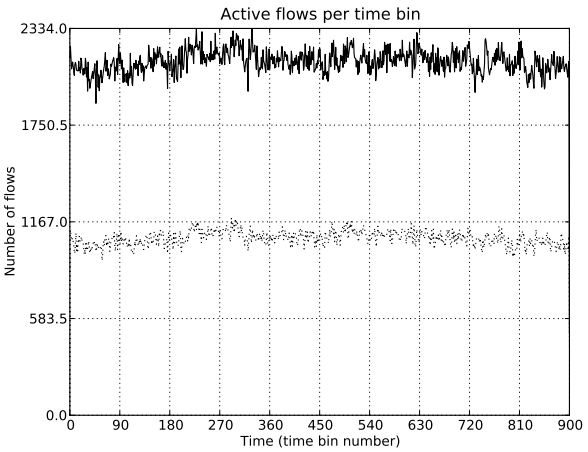
(c) /16 subnet-500 ms



(d) /16 subnet-1000 ms



(e) /24 subnet-500 ms



(f) /24 subnet-1000 ms

Fig. 6 Flow activity and persistence per aggregation level

5.2 Network traffic modeling

An accurate network traffic model is expected to capture the prominent traffic properties including short- and long range dependence, self-similarity in large time scale, and multifractality in short time scale. On the other hand, it has been observed that Internet traffic also exhibits non-stationary and non-linear properties. Therefore, in order to benchmark the given Routing Update Process (RUP) heuristics, one needs adequate traffic models fit and predict realistic IP backbone network traffic. This section summarizes the state-of-the-art of the network traffic models as will be used for experimental evaluation in Section 7.

Network traffic analysis studies in the last decades have uncovered the subtle pattern of *self-similarity* in network traffic time series. Stochastic self-similarity describes a statistical property of the traffic and manifests itself in several equivalent fashions: slowly decaying variance, long range dependence (LRD), non-degenerate autocorrelation, or Hurst effect. Intuitively, a process is said to be self-similar if its statistical behavior is independent of time-scale. Formally, a time series $Y = \{y_t | t \in T\}$ is self-similar if its autocorrelation function decays only hyperbolically instead of exponentially [28]. For self-similar processes, the autocorrelation function drops hyperbolically (not exponentially) toward zero but may never reach zero (non-summable autocorrelation function). The 'degree' of self-similarity is often denoted by the Hurst parameter, which is a measure of the persistence of a statistical phenomenon, denoting the length of the *long-range dependence* of a stochastic process.

After the seminal work reported in [16] confirmed mathematically in [28], it has been commonly accepted that Internet traffic behaves statistically self-similar [21, 10] and that aggregating streams of such traffic typically intensifies the self-similarity ("burstiness") instead of smoothing it.

5.2.1 State-of-the-art

Many studies have been conducted to predict the behavior of network traffic on time scales larger than a second. In this section, we give a short overview of these studies some of which also considered traffic prediction on a smaller time scale.

Fractional AutoRegressive Integrated Moving Average (FARIMA) models were successfully used in [25] to predict video, and Internet traffic on a timescale of one second or larger. The self-similar character of network traffic was shown to be adequately captured. However, at smaller time scales, depending on the specific traffic trace, it was also shown that the signal-to-noise (SNR) significantly decreases. As the FARIMA model cannot capture the multifractality which has been observed in the network traffic at short-time scale, [17] introduces the *AutoRegressive Integrated Moving Average* (*ARIMA*) - *Generalized Auto Regressive Conditional Heteroscedasticity (GARCH)* model. The resulting *ARIMA-GARCH* model is a non-linear time series model which combines the linear ARIMA model (with conditional mean but constant variance) with a conditional variance GARCH model (as will be further detailed in the next section). It captures both SRD and LRD traffic properties and observes self-similarity and multifractality. Results obtained using the ARIMA-GARCH model show that the predictive performance of the ARIMA-GARCH model outperforms traditional FARIMA model. Nevertheless, even if the prediction results can capture the actual traffic very well, some prediction errors can not be avoided because the real trace dynamic variance is out of the expectation of the GARCH model variance prediction.

In [11] wavelet neural networks proved to fit the self-similar structure of network traffic quite well too. Again, the authors concluded that predictions are dependent on the timescale – sub-second prediction performing consistently worse – and on the specific type of the network trace. The work presented in [26] analyzes the predictability of network traffic bit rates using *AutoRegressive Moving Average* (*ARMA*) and *Markov-Modulated Poisson Process* (*MMPP*) models. Under the assumption that those models are appropriate, authors of [26] developed analytic expressions to describe bounded predictability. Traffic aggregation and smoothing proved to monotonically increase the predictability of the network traffic. At last, the authors of [22] came to the same conclusion after performing a predictability analysis of large set of traffic traces. The work also shows that a decreasing bin size in aggregating traffic increases the variance of the resulting signal. This implies that short-term traffic evolution (which implies small bin sizes by definition) has a higher variability than longer-term traffic.

Based on the results of prior studies on short-term network traffic modeling, we will further focus on the ARIMA-GARCH model for our purposes. To ease the theoretical introduction, as well as to enable comparison with simpler models, in the next subsections we will introduce the ARMA model, the ARIMA model and the GARCH model separately.

5.2.2 AutoRegressive (Integrated) Moving Average (ARIMA) models

The *AutoRegressive Moving Average ARMA*(p,q) model is defined as follows:

$$y_t = \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{j=1}^{q} \beta_j w_{t-j} + w_t \tag{1}$$

This formula combines two techniques: i) autoregression, which reflects that a prediction is based on the signal itself (using p previous values) referring to the first term, and

ii) moving averages, reflected by q terms of the white noise series w_t (with $E(w_t) = 0$ and $Var(w_t) = \sigma^2$) which is put through a linear non-recursive filter determined by the coefficients α_i (weighted average). The autoregressive part directly reflects the *Short Range Dependence* (SRD) of a time series. Whereas these models have been successfully applied to model network traffic, e.g. [6], they are also known to be unable of modeling non-stationary time series. Stationarity implies that the probability distribution characterizing the time series mean and variance remains constant over time.

Some non-stationary time series can be made stationary by one or more levels of differencing[2]. Once the resulting differenced time series is stationary, an $ARMA(p,q)$ model can subsequently be fit and predictions can be made by integrating the predictions back. The resulting model of this approach is called the *AutoRegressive Integrated Moving Average* (ARIMA) model. The resulting model including the lag operator L[3] for $ARIMA(p,d,q)$ is defined as follows (d referring to the number of levels of differencing):

$$(1-\sum_{i=1}^{p}\alpha_i L^i)(1-L)^d y_t = (1+\sum_{j=1}^{q}\beta_j L^j)w_t \quad (2)$$

5.2.3 Generalized Autoregressive Conditional Heteroskedasticity traffic model

The ARIMA model cannot capture the multifractality which has been observed in some Internet traffic traces. For this reason, the Multifractal Wavelet model (MWM, [24]) has been introduced. However, while the MWM model can capture short timescale multifractality, it cannot predict traffic. For this purpose, the work of [32] introduces the ARIMA - Generalized AutoRegressive Conditional Heteroscedasticity (GARCH) model, a non-linear time series model which combines the linear ARIMA model (with constant variance) with conditional variance GARCH model [7]. The term "conditional" implies an explicit dependence of the variance of the noise/innovation series w_t from the ARIMA model on a past sequence of observations. The $GARCH(r,s)$ model for the conditional variance σ^2 of the innovation series w_t follows an $ARMA(p=r,q=s)$ model of the following form:

$$\sigma_t^2 = \sum_{i=1}^{r}\gamma_i\sigma_{t-i}^2 + \sum_{j=1}^{s}\omega_j w_{t-j}^2 + w_t \quad (3)$$

where, r and s are positive integers.

6 Packet loss reduction

Once a network traffic model is determined, which is able to adequately capture and predict network traffic behavior during the update of the routing table entries, we return to our initial target of reducing the resulting packet loss. For this purpose, we will formalize the concepts of "recovery time" of an updated routing table entry and the resulting "packet loss" during its update in this section.

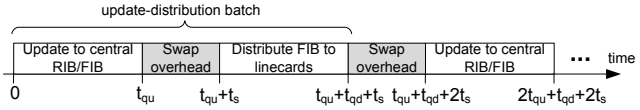


Fig. 7 Timeline of the router update process

6.1 Recovery time of an updated routing table entry

Given the batch structure of the router update process (see Section 2), all affected traffic flows directed to a given IP destination address are recovered when the following conditions are met: i) the IGP routing (table) entry for the prefix including that IP destination address is updated and stored in the central RIB, ii) the corresponding FIB entry is updated and stored in the central FIB, and iii) the batch number b_i that comprises this updated entry is distributed towards the LFIBs on the line cards. To simplify the description, we will consider that after its re-computation, the RIB and FIB updates are performed altogether by summing their update time (and thus refer to a RIB/FIB entry update). Thus, assuming a fixed batch size of x_u (number) of updated RIB/FIB entries and a given order of these entries, the recovery time of a flow f_i is characterized by the following formula:

$$r(f_i) = b_i x_u(t_u + t_d) + (2b_i - 1)t_s \quad (4)$$

Here, t_u refers to the update time for a single RIB/FIB entry, t_d to the distribution time for a single entry, and t_s to the swapping time interval between an update and a batch distribution operation. This process is represented in Figure 7.

The concept of recovery time can be illustrated by the following example. Assume that a network failure results into the disruption of 5000 traffic flows (F_{5000}) and that a single RIB/FIB entry is associated to each of their destination. If we use the following values $t_u = 100$ μs, $t_d = 100$ μs, $x_u = 100$, $t_s = 5000$ μs, this configuration results into 50 update-distribution batches, each taking $100 \times (100 + 100) + 5000 = 25000\mu$ s. Taking into account the additional intermediate swapping times between the batches, this results into $50 \times 25000 + 49 \times 5000 = 1495000$ μs, or 1.495 s to recover all flows.

The *Earliest Recovery Time (ERT)* possible for a given flow f_j, j referring to the waiting position (in the batch) of the updated RIB/FIB entry associated to this flow at time t, is defined as follows:

$$ERT(f_j,t) = t + j(t_u + t_d) + t_s \quad (5)$$

[2] A differenced time series y_t generates a new time series of differences $z_t = y_t - y_{t-1}$

[3] applying lag operator L on y_t generates y_{t-1}, and a power i of L defines a similar recursive process of the order i

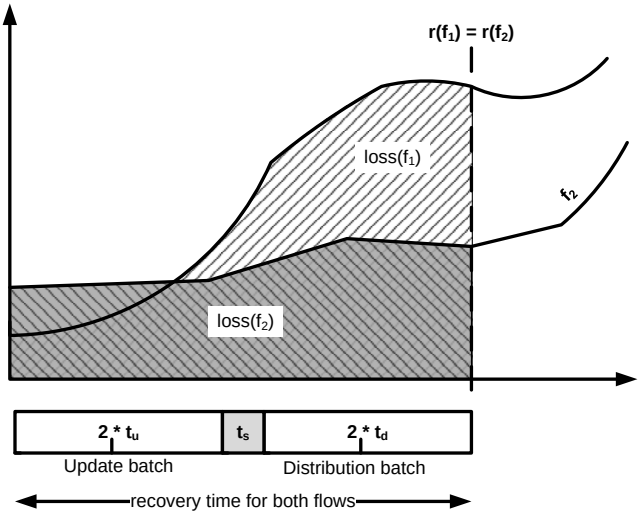


Fig. 8 Packet loss under dynamic traffic conditions

The ERT is the time interval of the smallest batch including the updated RIB/FIB entry associated to the considered flow together with the RIB/FIB entries updated before it. Indeed, in this situation only one swapping time interval is needed (separating the update and distribution batch) together with the time needed to update and distribute all prefixes comprised within the batch.

6.1.1 Packet loss induced by an updated routing entry

Packet loss occurs for network traffic flows as long as their corresponding RIB/FIB entries are not updated and distributed on line cards. For example, if two network traffic flows are recovered from updates part of the same batch, their recovery time is the same, and packet loss occurs for both flows up to the moment of recovery. This is represented in Figure 8. The resulting packet loss can be interpreted as the integral over time from the moment the failure happens (time 0 in the figure), up to the moment of recovery ($r(f_1) = r(f_2)$ in the figure). The loss resulting from the recovery of flow f_i relates to its bit rate, $br(f_i)$, through the following formula:

$$loss(f_i) = \int_0^{r(f_i)} br(f_i)dt \tag{6}$$

Packet loss calculation can be illustrated for the following simple example. Assume we have a small set F_4 consisting of four flows with associated constant bit rates $br(f_1) = 5, br(f_2) = 2, br(f_3) = 4, br(f_4) = 1$ (in Mbps), and $t_u = 100$ μs, $t_d = 90$ μs, $x_u = 2$, $t_s = 70$ μs, this results into a packet loss of $260 \times 10 + 590 \times 5 = 5550$ Kbps.

The above formula illustrates that the order in which RIB/FIB entries are updated, can heavily influence the resulting packet loss. Randomly updating RIB/FIB entries, as it is usually the case, can result into the situation where recovery of higher bit rate flows is delayed because RIB/FIB entries associated to lower bit rate flows are updated earlier. This delay results in high cumulative packet loss (for example f_3 is proportionally bigger than f_2, but however needs to wait until $f_1 \ldots f_2$ are updated).

6.2 Packet loss reduction heuristics

Assuming that we know the time series $br(f_i,t)$ of all flows (using traffic modeling techniques), in this section we formulate heuristics or packet loss reduction functions. Reduction of packet losses can be achieved by: i) updating the RIB/FIB entries associated to a (set of) flows in a different order, and/or ii) bundling the resulting updated RIB/FIB entries in well-specified consecutive batches. As such, an optimization function is defined as a function which maps a given traffic model (set of well-defined $br(f_i,t)$-functions) at a certain time t to a tuple $(flowordering, batching)$. The first part of the tuple denotes a permutation of the flows, the second part refers to a decomposition of the flows into ordered batches.

We evaluate two variants: a first heuristic which assumes fixed batch size, and thus only reduces packet loss by RIB/FIB entries reordering, and a second heuristic which works on both dimensions, i.e., reducing packet loss by changing both the RIB/FIB entries order and the set of batch sizes.

6.2.1 Fixed batch size

Given a fixed batch size of n entries and an adequate traffic model, we can exactly calculate the resulting recovery times for all batches. Once we know in which batch a routing entry is contained, we can calculate the associated packet loss. Therefore, the following approach can be applied.

Consecutively consider all batches to be processed, starting from the first batch, and apply the following iterative process:

1. Calculate the *recovery time for the current batch* of RIB/FIB entries
2. For *every routing table entry* corresponding to flow f_j that is not yet updated, calculate the *resulting packet loss* if its corresponding RIB/FIB entry would be contained in the *current batch* of RIB/FIB entries
3. *Sort* all resulting packet losses in decreasing order
4. *Select* the n entries with the highest associated packet loss and their associated RIB/FIB entry
5. *Terminate the current batch*, and remove the recovered flows from the working set
6. Continue the process from step 1 for the next batch of RIB/FIB entries and the remaining flows.

6.2.2 Variable batch size

Building further on the idea of the heuristic described in [29], we can formulate a dynamic variant which takes into account both the flow ordering (thus the order of RIB/FIB entries), and the batch size so as to minimize the packet losses. For this purpose, consider the following scenario. We denote by $F_n = f_1, \ldots, f_n$ the set of traffic flows affected by the failure, and by $b_{current} = (f_i, \ldots, f_{i+s})$ the set of flows for which the corresponding entries still need to be updated on routers' LFIB[4]. In this context, we have two possible options when in the middle of the ordered process of updating the router's LFIB entries (associated to the set of flows F_n) within our current batch of RIB/FIB entries $b_{current}$:

1. *Extension*: extend the current batch with the RIB/FIB entry associated to the next flow f_{i+s+1};
2. *Splitting*: terminate the current batch and put the RIB/FIB entry associated to the next flow into a new update-distribution batch.

We can now compare the additional cost of extension (ec) versus the additional cost of finishing (fin) the update-distribution batch to guide us into the decision above. The idea of the trade-off is illustrated in the simple example with flows to be updated as shown in Figure 9. Given a current batch (starting from the first one), the heuristic makes a trade-off between terminating the current batch before the next RIB/FIB entry is inserted into the current batch, or extending the current batch with the next RIB/FIB entry. For example, let's assume our current batch contains the RIB/FIB entries for the flows f_1, f_2 and f_3. According to the iterative character of the strategy, this means that extending the batch with the entries for the flows f_2 and f_3 had lower cost than splitting. In a next step, we can now either again extend the current batch with the RIB/FIB entry for the flow f_4 or split this batch and insert the entry for f_4 in a new update-distribution batch. Figure 9 shows the difference between both decisions: an extension delays the recovery time for f_1, f_2 and f_3, while a split delays the recovery time for f_4 (and all flows following f_4). Table 1 quantitatively shows the recovery times for all flows in both scenario's and compares them with the earliest recovery time (ERT) possible per flow. The decision $ec < fin$ will compare the difference in recovery time multiplied with the associated bandwidth to compare packet loss of both options.

We can now further formalize the trade-off using the following definition for the extension cost $ec(b_{current})$ for this batch, with $t_{bcurrent}$ as the starting time of the current batch $b_{current}$:

$$ec(b_{current}) = ec_{is} = \sum_{j=i}^{i+s} \int_a^b br(f_j, t)dt \tag{7}$$

with

$$a = ERT(f_j, t_{bcurrent}) \tag{8}$$

and

$$b = ERT(f_{i+k}, t_{bcurrent}) + (i+s-j+1)(t_u + t_d) \tag{9}$$

The formula expresses the fact that, by extending the current batch, the recovery time of every RIB/FIB entry comprised in the current batch will result into an additional delay compared to the minimal delay it can experience (given the position of that entry in the current batch). The minimal delay an entry can experience is determined by the ERT, i.e., the time elapsing for an entry positioned as the last one in an update quantum having no earlier update quanta. This additional delay, when multiplied with the associated bit rate, allows deducing the additional loss caused by the extension of the update-distribution batch. For example, if the RIB/FIB entry associated to the first flow f_i was already delayed by s update times t_u (as this entry was not directly distributed but put in the same batch as the s next entries ranging from i to $i+s$), extending the current batch by one element (to reach $i+s+1$ elements) further delays the recovery time of the entry i. On the contrary, the recovery of the last entry $i+s$ of the current batch will only be delayed by one update time t_u in case of extension of the current batch.

On the other hand, terminating the current batch has also an associated cost, as it will introduce additional delay for the recovery of coming flows, resulting from the additional swapping cost. This termination condition can be formulated as follows:

$$fin_{b_{current}} = \sum_{f_j \notin b_{current}} \int_{ERT(f_j, t_{bcurrent})}^{ERT(f_j, t_{bcurrent})+2t_s.} br(f_j, t)dt \tag{10}$$

Our configuration strategy now consists in identifying the action with the least associated cost. The overall algorithm can be expressed as follows:

1. *Add all affected routing table entries* to the working set
2. *Sort* all entries in the working set in decreasing order of their current cumulative packet loss $loss(f_j, t_{current})$
3. *Compute both extension and splitting cost:*
   - If no current batch exists (RIB/FIB entry associated to the first flow), then *create a batch* and add this entry into this newly created batch.

[4] The RIB/FIB entries for the prefixes corresponding to the flows prior to f_i have already been updated in an ordered manner (from f_1 to $f_{(i-1)}$)

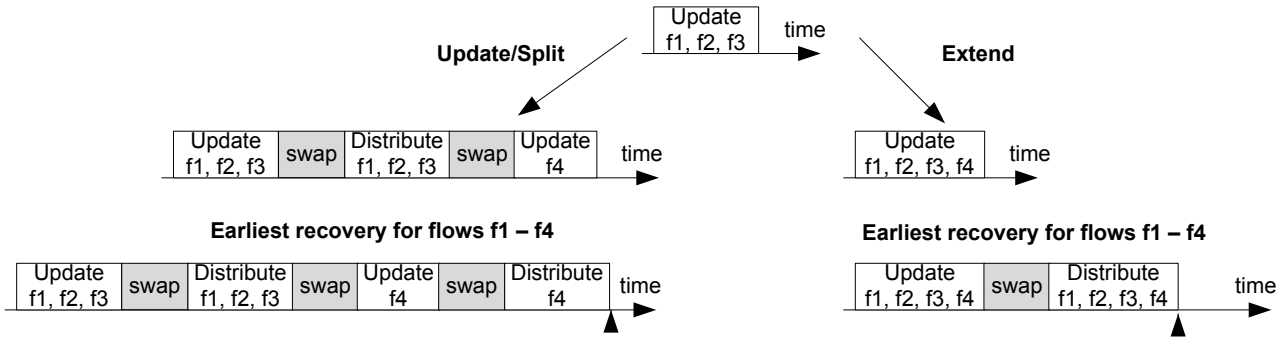


Fig. 9 Compare extension vs. split-scenario for F_4

	Earliest recovery time	Update/split scenario		Extension scenario	
		Recovery time	Diff. with earliest recovery time	Recovery time	Diff. with earliest recovery time
f_1	$t_u+t_d+t_s$	$3t_u+3t_d+t_s$	$2(t_u+t_d)$	$4t_u+4t_d+t_s$	$3(t_u+t_d)$
f_2	$2(t_u+t_d)+t_s$		(t_u+t_d)		$2(t_u+t_d)$
f_3	$3(t_u+t_d)+t_s$		0		(t_u+t_d)
f_4	$4(t_u+t_d)+t_s$	$4t_u+4t_d+3t_s$	$2t_s$		0

Table 1 Recovery time comparison of F_4-scenario: split vs. extension

- Otherwise:
  - If the extension cost is smaller than the splitting cost, then *add* the corresponding RIB/FIB entry in the sorted list to the *current batch*;
  - Otherwise, *create a new batch* and add the corresponding RIB/FIB entry into this newly created batch.
4. *Remove* the added flow from the working set
5. *Repeat* the procedure from step 2 until the working set is empty

7 Experimental results

In the previous section theoretical foundations have been developed to model the IP router update process and the network traffic through the router. Both aspects were used to formulate two heuristics enabling packet loss reduction during the update of the LFIB entries (process realized via the update of the corresponding RIB/FIB entries). In this section, real network traces will be used to evaluate the performance gain (i.e., packet loss decrease) obtained by means of the proposed heuristics. First, we detail the overall methodology and experimental environment, then we benchmark the discussed network models of Section 5.2. At last, we measure the overall gain of the suggested approach in terms of resulting packet loss reduction.

7.1 Environment and methodology

As indicated in Section 5, a set of PCAP traces obtained in December 2009 was taken from the MAWI project [9]. These traces were preprocessed using Python scripting at three time interval levels (100 ms, 500 ms and 1000 ms), referred to as time bins, and at three spatial levels (/8, /16 and /24 subnetworks) with respect to their aggregated traffic volume, i.e., aggregated packet sizes per time-bin per prefix. The choice for this set of spatial aggregation levels allows us to set an upper and lower bound on the possible gain that could be achieved, because the typical aggregation level in usual routing operations (especially when distributed to inter-domain environments) is uncommon to be above /8 or below /24. Next, the resulting traffic time series were analyzed and fit to one of the described traffic models using R software [23]. The resulting modeled data was used as input to the heuristic implemented in C++/Python. All the experiments ran on a regular desktop computer equipped with an AMD Athlon 2.7 GHz CPU and 4 GB RAM. The entire process is shown in Figure 10. To obtain representative results, all experiments were repeated 1000 times, and the resulting values were averaged.

7.2 Network traffic fitting and prediction

The goal of the first experiment is to validate fitting abilities of network traffic modeling techniques introduced in Section 5.2 with respect to the given set of MAWI trace files. We evaluated the following variants: i) `PERSIST`, ii) `ARMA`, iii) `ARIMA`, iv) `ARMA-GARCH`, and v) `ARIMA-GARCH`. The first model represents the naive assumption that network traffic

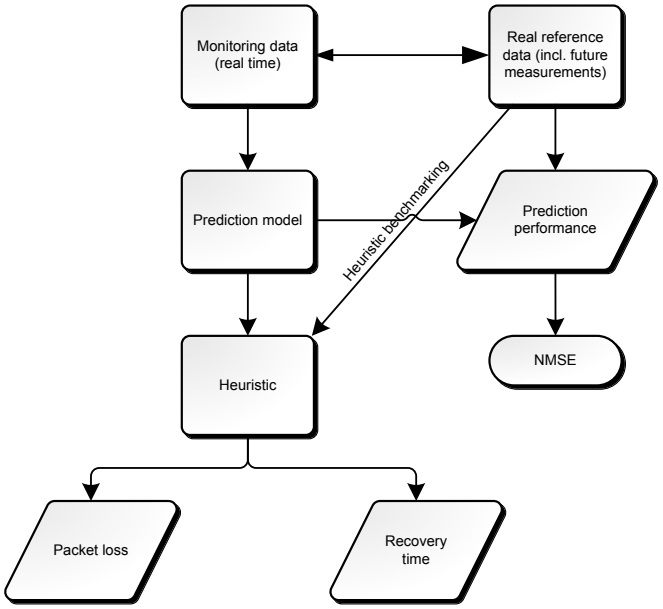


Fig. 10 Benchmarking process

for a given time bin will behave exactly the same as the previous time bin (see Section 5.1, as in [29]). The latter models refer to those described in Section 5.2, and have configurable parameters and different orders determined by their parameters p, q, d and r, s.

The main task in automatic ARIMA forecasting is to select an appropriate model order. To automate the process of optimal parameter selection, we used the order selection strategy of [14] for ARMA and ARIMA models restricted to maximum values of 10. The strategy selects those parameters which minimize the Akaike's Information Criterium (AIC) among all fitting models. Once the best parameters are selected, Maximum Likelihood Estimators (MLE) are computed to deduce the best fitting coefficients for the ARMA and ARIMA models [14]. The lowest value for the d parameter of the ARIMA model, resulting into a stationary time series, according to the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for stationarity[5], is then selected [14].

For the GARCH model, the values of parameters $r = 1$ and $s = 1$ were chosen to reduce the resulting computational complexity, as well as to avoid non-optimizable parameter values due to the lack of sufficient information[6]. The GARCH$(1,1)$-coefficients were estimated using the Quasi-Maximum Likelihood Estimator (QMLE) [8].

For one MAWI trace, the fitting procedure at all aggregation levels takes about 20 hours on the referred computer. However, in the context of the router update, this process can be executed in background. Figure 11 shows the average Normalized Mean Square Errors (NMSE) of fitting the models to the set of MAWI traces. The NMSE is defined as follows[7].

$$NMSE = \frac{\sum_n (\widehat{y}(x) - y(x))^2}{\sum_n (y(x) - \mu_T)^2} \tag{11}$$

The NMSE allows to measure how much of the variance of a signal can be accommodated by a model. Smaller NMSE values correspond to better fitting models. Figure 11 illustrates the obtained average[8] fitting errors obtained for the set of referred MAWI traces.

Figure 11a shows the error for the models at three different bin sizes (100, 500 and 1000 ms). Figure 11b depicts the error with respect to different spatial aggregation levels (i.e. subnet lengths): /24, /16 and /8 subnets.

As might be expected, the persistence model has a significantly larger fitting error than the four other models. In general the error is at typically twice as high. Not unexpected either, the ARIMA-GARCH$(1,1)$ model gives the best performance on average on all aggregation levels, followed by – in decreasing order of performance – ARMA-GARCH$(1,1)$, ARIMA, and ARMA model. However, the fitting performance of all more advanced strategies (2 to 5) is rather similar, and differences are small. This result may be understood from the fact that sub-second dynamics of network traffic in IP backbones is very high, resulting into hard fitting and prediction problems. This result also corroborates those obtained by other studies (see state-of-the-art in Section 5.2). More aggregation, either by using larger time bins or by using smaller subnetworks (larger network masks or netmasks), leads to better fitting models; however, the relative gain between the techniques does not change drastically. Clearly, more aggregated traffic resulting into smoother and more predictable network traffic (also see Figure 4), which explains that the fitting error using /8 subnet levels is significantly less compared to the ones obtained using /16 or /24 subnets for spatial aggregation.

7.3 Packet loss evaluation

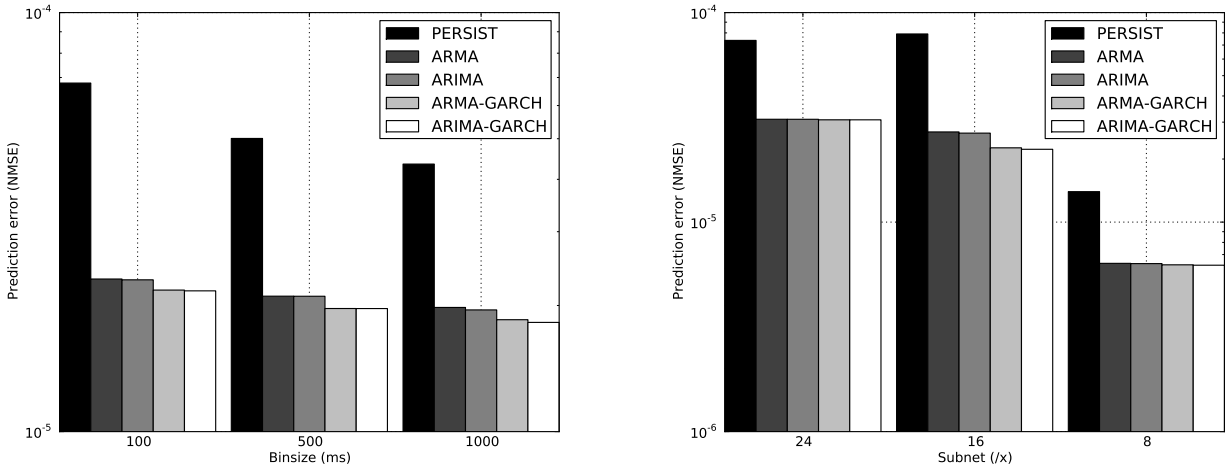
Given the traffic models of the previous subsection, the goal of the experiments below is to measure the combined gain in resulting packet loss between a set of combinations of traffic models and heuristics described in Section 6.2. The following parameters were fixed: the update time t_u needed for one RIB/FIB entry: 100μs, the distribution time t_d towards the LFIB's: 100μs and the swapping time $t_s = 5000\mu$s. For this experiment we assumed that a failure affected 5000 randomly chosen prefixes (if available) from a given MAWI

[5] The KPSS test assesses the null hypothesis that a univariate time series trend is stationary against the alternative that it is a non-stationary unit-root process

[6] Not all time series corresponding to the volumes of a given prefix are dense enough. Some series have more zero values than others, resulting into non-optimizable parameters due to singular Hessian matrices.

[7] $\widehat{y}(x)$ referring to the predictions, $y(x)$ referring to the actual values

[8] over all flows in all trace files



(a) Fitting error of traffic model vs. bin size

(b) Fitting error of traffic model vs. subnet length

Fig. 11 Fitting error of different traffic models

trace. From the resulting routing table update event, packet loss and recovery time was calculated. This procedure was repeated 1000 times to obtain representative average results.

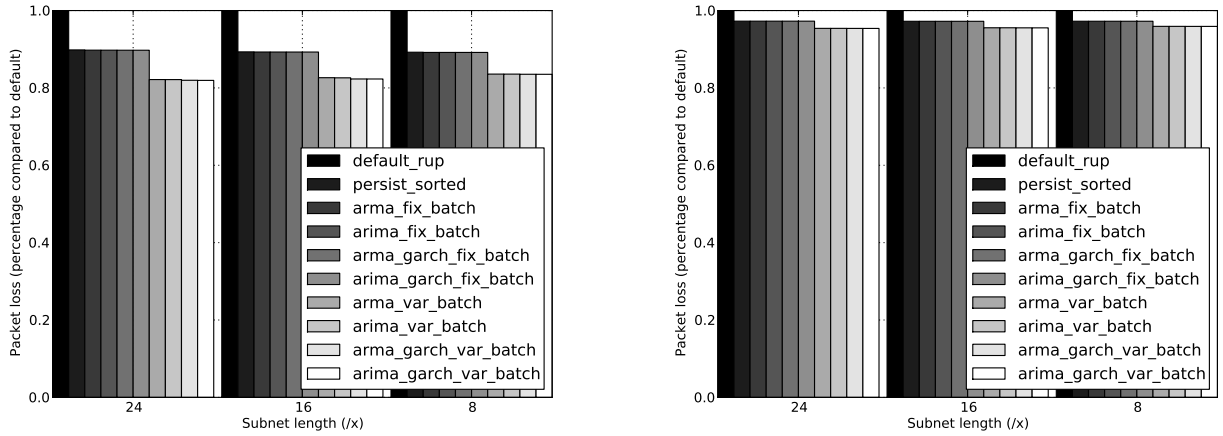
Average results obtained by the evaluation of the resulting decrease in packet loss, is depicted in Figure 12. The left part of the figure (Figure 12a) shows the resulting packet loss vs. the length of subnets using 100 ms time bins, while the right part (Figure) depicts the packet loss vs. several subnet lengths using 500 ms time bins. The following combinations were evaluated:

1. The default traffic-agnostic router update as described in Section 2 with a typical fixed batch size of 100 entries (`default_rup`).
2. The routing update process as defined in [29], assuming persistent network traffic, sorting the resulting routing entries and dynamically updating the batches (`persist_sorted`).
3. The heuristic using fixed batch sizes from Section 6.2.1 in combination with: i) the ARMA models (`arma_fix_batch`), ii) the ARIMA models (`arima_fix_batch`), iii) the ARMA-GARCH models (`arma_garch_fix_batch`) or the iv) ARIMA-GARCH models (`arima_garch_fix_batch`) as fit in Section 7.2. The fixed batch size was also set to 100 prefixes.
4. The heuristic enabling variable batch sizes from Section 6.2.2 in combination with: i) the ARMA models (`arma_var_batch`), ii) the ARIMA models (`arima_var_batch`), the iii) ARMA-GARCH models (`arma_garch_var_batch`) or the iv) ARIMA-GARCH models (`arima_garch_var_batch`) as fit in Section 7.2.

Combinations making use of fixed batch sizes (combination 1, and 3 to 6) were evaluated with 100 entries in a batch ($x_u = 100$). This is a representative choice based on [13] (also see Section 2). Smaller batch sizes typically result into relatively too large swapping times (discouraging multiple batches and swapping), and bigger batch sizes result into lower optimization potential as illustrated in the results of [29].

Figure 12a illustrates that: i) on average, a decrease of 18 percent packet loss can be obtained using the variable batch size heuristic with any of the proposed prediction techniques compared to the default random routing table update, and ii) about 8 percent packet loss compared to the more obvious optimization technique `persist_sorted` based on [29] (for the case of using /24 subnets with 100 ms time bins). These results give a more realistic view on the decrease in packet loss one could obtain during routing table updates compared to those in [29] (which reports up to 80 percent decreases). This validates that the high dynamicity of network traffic in small time scales makes the problem of traffic-driven routing table updates drastically more difficult than one could imagine at first sight.

Whereas it is obvious that the default behavior of updating the RIB/FIB entries for IGP prefixes in random order is constant, independently of the traffic prediction model, it is surprising that the specific choice of the prediction model –in combination with the optimization heuristic– has relatively low influence on the decrease in packet loss. This observation may be explained from the fact that the difference between NMSE of the prediction models is rather small (except for the persistence model, see Section 7.2), due to the fact that sub-second network traffic modeling is a hard problem.



(a) Packet loss decrease vs. default process using 100 ms time bins

(b) Packet loss decrease vs. default process using 500 ms time bins

Fig. 12 Average packet loss decrease vs. default routing table update process at different traffic aggregation levels

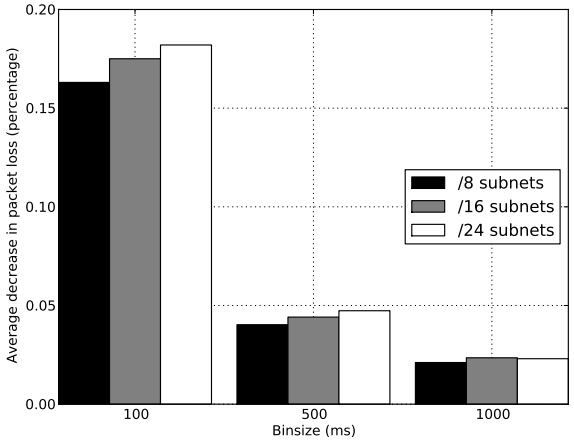


Fig. 13 Average packet loss decrease vs. default process over all aggregation levels

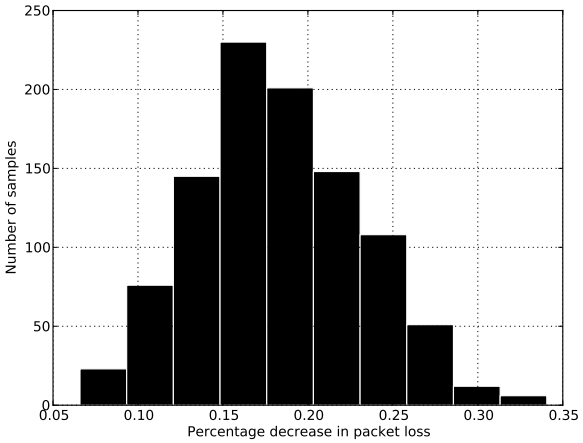


Fig. 14 Distribution of decrease in packet loss (/24 subnet prefixes and bin size 100 ms)

Figures 12a and 12b clarify the influence of traffic aggregation on the obtainable decrease in packet loss. Whereas larger aggregation levels improve the correctness of network traffic models (smoother and more equal traffic improves predictability, see Section 7.2), it also decreases the gain potential in packet loss compared to: i) `default_rup`) and ii) (`persist_sorted`). The more similar traffic behaves, the less difference RIB/FIB reordering makes. The following may be observed: smaller aggregation levels result into smaller packet loss decreases. Whereas in the best case, average reductions of about 18 percent are possible for /24 subnets in combination with 100 ms time bins, only reductions of about 5 percent decrease in packet loss were possible in our simulation environment when using 500 ms time bins. The average gain even becomes even smaller for 1000 ms time bins as indicated in Figure 13.

Our results show that on average, significant decreases in packet loss can be obtained. It is nevertheless important to determine the worst case: is it possible that the default random update shows in some case(s) lower packet loss than the combination of prediction and the optimization heuristic. This question is answered by the results obtained in Figure 14. This figure illustrates that a normal distribution was obtained around the mentioned averages in packet loss, with minimal and maximal improvements of respectively 5 and 35 percent in decrease of packet loss compared to the default routing table update. We found no executions where the default behavior was better than the suggestion solution. Similar distributions were found at other temporal and spatial aggregation levels for which the average decrease of packet loss is as depicted in Figure 13. The results illustrate that

using a prediction model in combination with the described heuristic makes most sense at low aggregation levels, where up to 18 percent (average) decrease in packet loss can be achieved.

7.4 Recovery time evaluation

The packet loss reduction heuristics from Section 6 influence the resulting global recovery time of all affected traffic flows (the duration of the entire routing table update). This is because packet loss decreases are achieved by reordering RIB/FIB entries and by online adapting update-distribution batch sizes. The latter results into more or less batches and swapping events, affecting the resulting global recovery time. In general, smaller resulting packet losses do not necessarily imply that the total recovery time will also be decreased. Depending on the specific traffic trace, splitting batches can be beneficial with respect to the resulting packet loss (lower splitting cost vs. extension cost, see Section 6.2.2). More batches lead to more swapping times, resulting into larger recovery times.

Figure 15 shows the obtained average global recovery times for the different combinations of traffic models and packet loss reduction heuristics. The figure depicts the average recovery times using modeled traffic at bin sizes of 100 ms (left) and 500 ms (right) at three spatial aggregation levels (/24, /16 and /8 subnets). Strategies 1 and 3 to 6 use fixed batch sizes, and thus result into the same fixed recovery time (only the first is depicted). If traffic is aggregated into /8 subnets, there are not as many aggregated subnet prefixes (max 256). Therefore, the resulting average recovery time is of a lower order than the ones using /16 or /24 subnets. This figure further illustrates that higher traffic aggregation levels typically result into recovery times that are closer to the ones obtained by fixed batch strategies because there remains less opportunity to minimize packet loss by reordering RIB/FIB entries or dynamically resizing batches (see previous section). From Figure 15a one might observe that recovery times of packet loss reduction strategies on average result into shorter recovery times ranging from 30 percent faster, to similar recovery times as update strategies using fixed batch sizes. Figure 15b illustrates that using packet loss reduction heuristics with less fine-grained traffic models (i.e. using 500 ms bin sizes) further reduce the difference with fixed batch update strategies, resulting into average recovery times between 10 percent less, to similar recovery times as fixed batch size strategies.

8 Computational cost of the procedure

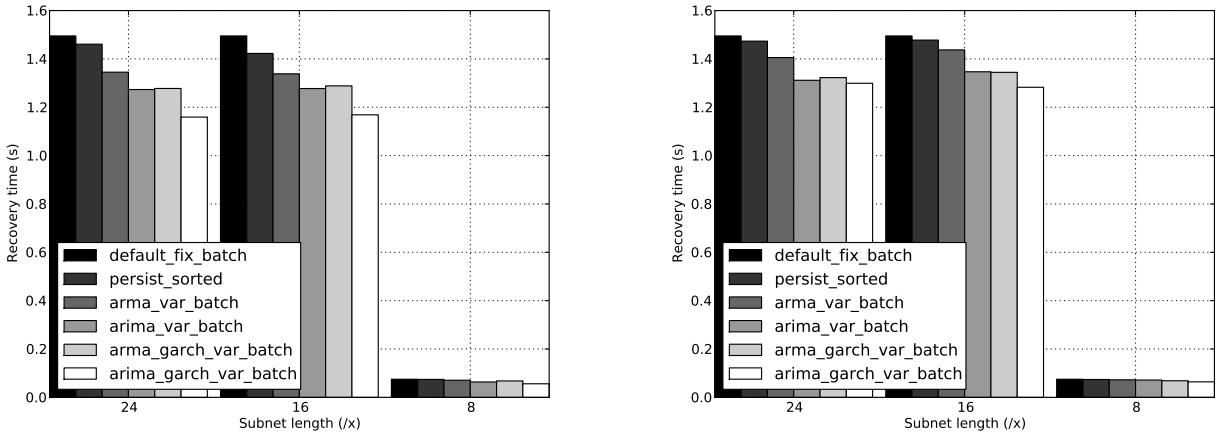
The execution of the proposed procedure can be classified as relatively intensive from a computational perspective. However, many parts can run offline or in background. Fitting ARMA and ARIMA models to time series samples have a computational complexity which is bounded by $O(m^3T)$, where T is the length of the time series sample, and $m = max(p,q+1)$. Fitting GARCH models reduces to hard non-linear optimization problems. To the knowledge of the authors there is no clear bound to the computational complexity of fitting these models for arbitrary (r,s) values.

The computation of the described optimization heuristics involves operations which are bounded by *O(n log n)* during every step split/extension-step, caused by the need for real-time sorting. While this results in a heavy procedure, it can be drastically reduced by using hardware-implemented sorting algorithms which reduce to constant time sorting for a given number of values [30].

9 Conclusion

Link-state routing protocols used within current routing domains can have re-convergence times in the order of seconds when topology changes occur. During these periods of network recovery, affected network traffic is lost as long as their corresponding routing entries are not updated. We showed that first updating routing entries corresponding to high bit rate network traffic is not straightforward because network traffic can highly fluctuate during these periods of re-convergence. In this paper, we proposed a local router mechanism to reduce packet loss during re-convergence periods. This mechanism works independently of other routers and can be applied upon local failure detection as well as upon reception of topology change messages (link-state updates) received from non-adjacent routers.

Our approach used AR(I)MA-GARCH network traffic models to capture sub-second traffic fluctuations significantly better than a persistence traffic model assuming stable traffic during re-convergence. As a result, we showed that we can reduce a clear amount of packet loss resulting from rerouting events, depending on the used traffic model, the used heuristic and the aggregation level at which the traffic is modeled. We obtained packet loss reduction results varying from 18 percent on /24 subnetworks using 100 ms time bins to 2 percent using 1000 ms time bins. Figure 14 provided evidence that in some cases even larger gains could be obtained. The application of the used techniques also showed that the resulting recovery times over all affected routing entries are not higher than when using standard router update mechanisms.



(a) Average recovery time vs. default process using 100 ms time bins

(b) Average recovery time vs. default process using 500 ms time bins

Fig. 15 Average recovery time of dynamic vs. fixed batching strategies

Acknowledgment

This work is supported by the European Commission (EC) Seventh Framework Programme (FP7) ECODE project (Grant nr. 223936).

References

1. Alcatel 7750 SR OS router configuration guide. `http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config5.html`
2. Interface Statistics on JUNOS 9.2. `http://www.juniper.net/techpubs/software/junos/junos91/swcmdref-basics-services/monitor-interface.html`
3. IS-IS Support for Priority-Driven IP Prefix RIB Installation. `http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/fslocrib.html`
4. Passive measurement and analysis by national laboratory for applied network research project (nlanr). `http://www.nlanr.net/PMA/`
5. Statistics Service Commands on Cisco IOS-XR Software. `http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r2.0/system_management/command/reference/3yrstats.html`
6. Basu, S., Mukherjee, A., Klivansky, S.: Time series models for internet traffic. In: INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, vol. 2, pp. 611 –620 vol.2 (1996). DOI 10.1109/INFCOM.1996.493355
7. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics **31**(3), 307–327 (1986)
8. Bollerslev, T., Wooldridge, J.: Quasi-maximum likelihood estimation and inference in dynamic models with time-varying covariances. Econometric Reviews **11**(2), 143–172 (1992)
9. Cho, K., Mitsuya, K., Kato, A.: Traffic data repository at the WIDE project. In: ATEC '00: Proceedings of the annual conference on USENIX Annual Technical Conference, pp. 51–51. USENIX Association, Berkeley, CA, USA (2000)
10. Crovella, M.E., Bestavros, A.: Self-similarity in World Wide Web traffic: evidence and possible causes. IEEE/ACM Trans. Netw. **5**(6), 835–846 (1997). DOI http://dx.doi.org/10.1109/90.650143
11. Di, C., Hai-Hang, F., Qing-jia, L., Chun-xiao, C.: Multi-scale Internet Traffic Prediction Using Wavelet Neural Network Combined Model. In: Communications and Networking in China, 2006. ChinaCom '06. First International Conference on, pp. 1–5 (2006). DOI 10.1109/CHINACOM.2006.344786
12. Estan, C., Keys, K., Moore, D., Varghese, G.: Building a better NetFlow. SIGCOMM Comput. Commun. Rev. **34**(4), 245–256 (2004). DOI http://doi.acm.org/10.1145/1030194.1015495
13. Francois, P., Filsfils, C., Evans, J., Bonaventure, O.: Achieving sub-second IGP convergence in large IP networks. ACM SIGCOMM Computer Communication Review **35**(3), 33–44 (2005)
14. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. Monash Econometrics Working Papers 6/07, Monash University, Department of Econometrics (2007)
15. Katz, D., Ward, D.: Bidirectional Forwarding Detection. Internet-Draft draft-ietf-bfd-base-08, Internet Engineering Task Force (2008). Work in progress
16. Leland, W.E., Willinger, W., Taqqu, M.S., Wilson, D.V.: On the self-similar nature of Ethernet traffic. SIGCOMM Comput. Commun. Rev. **25**(1), 202–213 (1995). DOI http://doi.acm.org/10.1145/205447.205464
17. Marian, I., Dadarlat, V., Iancu, B.: A comparative study of the statistical methods suitable for network traffic estimation. In: ICCOM: Proceedings of the 13th WSEAS international conference on Communications, pp. 99–104. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2009)
18. McQuillan, J.M., Richer, I., Rosen, E.C.: An overview of the new routing algorithm for the ARPANET. SIGCOMM Comput. Commun. Rev. **25**(1), 54–60 (1995). DOI http://doi.acm.org/10.1145/205447.205453
19. Moy, J.: OSPF Version 2. RFC 2328, Internet Engineering Task Force (1998)
20. Narváez, P., Siu, K.Y., Tzeng, H.Y.: New dynamic algorithms for shortest path tree computation. IEEE/ACM Trans. Netw. **8**(6), 734–746 (2000). DOI http://dx.doi.org/10.1109/90.893870
21. Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. Networking, IEEE/ACM Transactions on **3**(3), 226–244 (1995). DOI 10.1109/90.392383
22. Qiao, Y., Skicewicz, J., Dinda, P.: An Empirical Study of the Multiscale Predictability of Network Traffic. In: HPDC '04: Proceedings of the 13th IEEE International Symposium on High Perfor-

mance Distributed Computing, pp. 66–76. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/HPDC.2004.3
23. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2009). ISBN 3-900051-07-0
24. Riedi, R., Crouse, M., Ribeiro, V., Baraniuk, R.: A multifractal wavelet model with application to network traffic. Information Theory, IEEE Transactions on **45**(3), 992 –1018 (1999). DOI 10.1109/18.761337
25. Sadek, N., Khotanzad, A., Chen, T.: ATM dynamic bandwidth allocation using F-ARIMA prediction model. In: Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on, pp. 359–363 (2003). DOI 10.1109/ICCCN.2003.1284194
26. Sang, A., qi Li, S.: A predictability analysis of network traffic. In: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 1, pp. 342–351 vol.1 (2000). DOI 10.1109/INFCOM.2000.832204
27. Shand, M.: IP Fast Reroute Framework. Internet-Draft draft-ietf-rtgwg-ipfrr-framework-08, Internet Engineering Task Force (2008). Work in progress
28. Taqqu, M.S., Willinger, W., Sherman, R.: Proof of a fundamental result in self-similar traffic modeling. SIGCOMM Comput. Commun. Rev. **27**(2), 5–23 (1997). DOI http://doi.acm.org/10.1145/263876.263879
29. Tavernier, W., Papadimitriou, D., Colle, D., Pickavet, M., Demeester, P.: Optimizing the IP router update process with traffic-driven updates. In: DRCN 2009. Washington D.C. (2009)
30. Tseng, Y.H., Wu, J.L.: On a constant-time, low-complexity winner-take-all neural network. Computers, IEEE Transactions on **44**(4), 601 –604 (1995). DOI 10.1109/12.376175
31. Wallerich, J., Dreger, H., Feldmann, A., Krishnamurthy, B., Willinger, W.: A methodology for studying persistency aspects of internet flows. SIGCOMM Comput. Commun. Rev. **35**(2), 23–36 (2005). DOI http://doi.acm.org/10.1145/1064413.1064417
32. Zhou, B., He, D., Sun, Z.: Network Traffic Modeling and Prediction with ARIMA/GARCH (2005)